CS 134

Physics

Game Engine Architecture
Chapter 12.1 – 12.3

Today in Video Games

Want to run Vulkan on iOS and macOS?

MoltenVK

Move to the next-generation, cross-platform, *Vulkan* graphics API on *iOS* and *macOS*. Build portable graphics applications and games using the modern, industry-standard *Vulkan* graphics API, and seamlessly run your application or game across many industry platforms, including *iOS* and *macOS*.

Vulkan is the graphics industry's new standard for predictable, high-performance graphics, providing you with unprecedented control of your graphics and compute pipelines. *Vulkan* is an open standard, developed by the *Khronos Group*, an industry consortium dedicated to the creation of open standards for the graphics industry. *Vulkan* is supported by a large number of major industry participants, including hardware vendors, driver implementers, and tool vendors, across many hardware and operating system platforms.

Homework

Median		110%	112%	109%
Std. Deviation		5%	38%	40%
100% percentile		10%	19%	41%

Physics

- So far, we have three classes:
  - Background
    - 2D array of tiles
  - Sprites
    - Position, animation frame
  - Camera
    - Position

Physics

- So far, we have three classes:
  - Background
    - 2D array of tiles, **shape**
  - Sprites
    - Position, **velocity**, animation frame, **shape**
  - Camera
    - Position, **shape**
- For shapes to make sense, you must make sure all the values you have have geometric meaning.

Physics


02005199 LV 033
×02
ENCOUNTERED AN ASSAILANT.
PHYSICS
BE ATTITUDE FOR GAINS...
1. DOT PRODUCT
2. SEPARATING AXIS
3. BE PRAYING

Physics

- Want to run at a fixed framerate
  - Faster than graphics
  - Often between 50fps and 200fps
- Handles Motion, Collision detection, collision resolution
  - Motion: Moving objects by their velocity
  - Collision Detection: Detecting overlaps
  - Collision Resolution: Handling overlaps

Physics

```
// Physics runs at 100fps, or 10ms / physics frame
int physicsDeltaMs = 10;
int lastPhysicsFrameMs;

// The game loop
while (!shouldExit) {
    // ...

    // Physics update
    do {
        // 1. Physics movement
        // 2. Physics collision detection
        // 3. Physics collision resolution
        lastPhysicsFrameMs += physicsDeltaMs;
    } while (lastPhysicsFrameMs + physicsDeltaMs < curFrameMs );

    // Normal update logic
    // …
}
```

Physics – Motion

- Most objects do not get moved here, simulation heavy objects only
- Vehicles: Cars, spaceships, etc
- Physics objects: Things with springs, mass, etc.

Physics – Motion

- Each object must have a velocity and mass in its state
- The “normal” tick() just sets its velocity.

- During physics tick, the object moves based on its velocity
- During physics collision resolution, resolve collisions based on physics equations
  - F=ma
  - Conservation of momentum

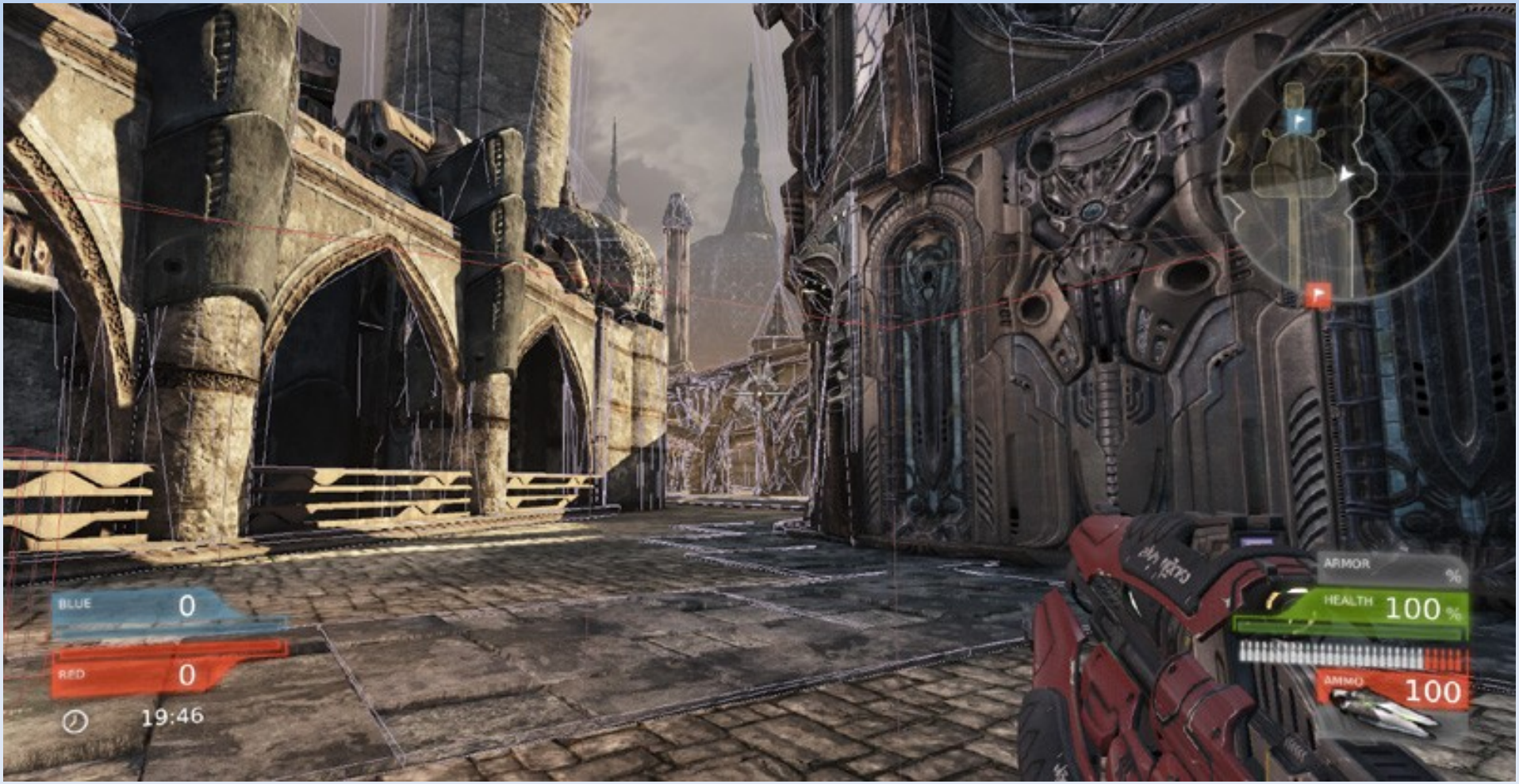
Physics – Collision Detection

- Remember the three slowest things?
  - Graphics, Physics, AI

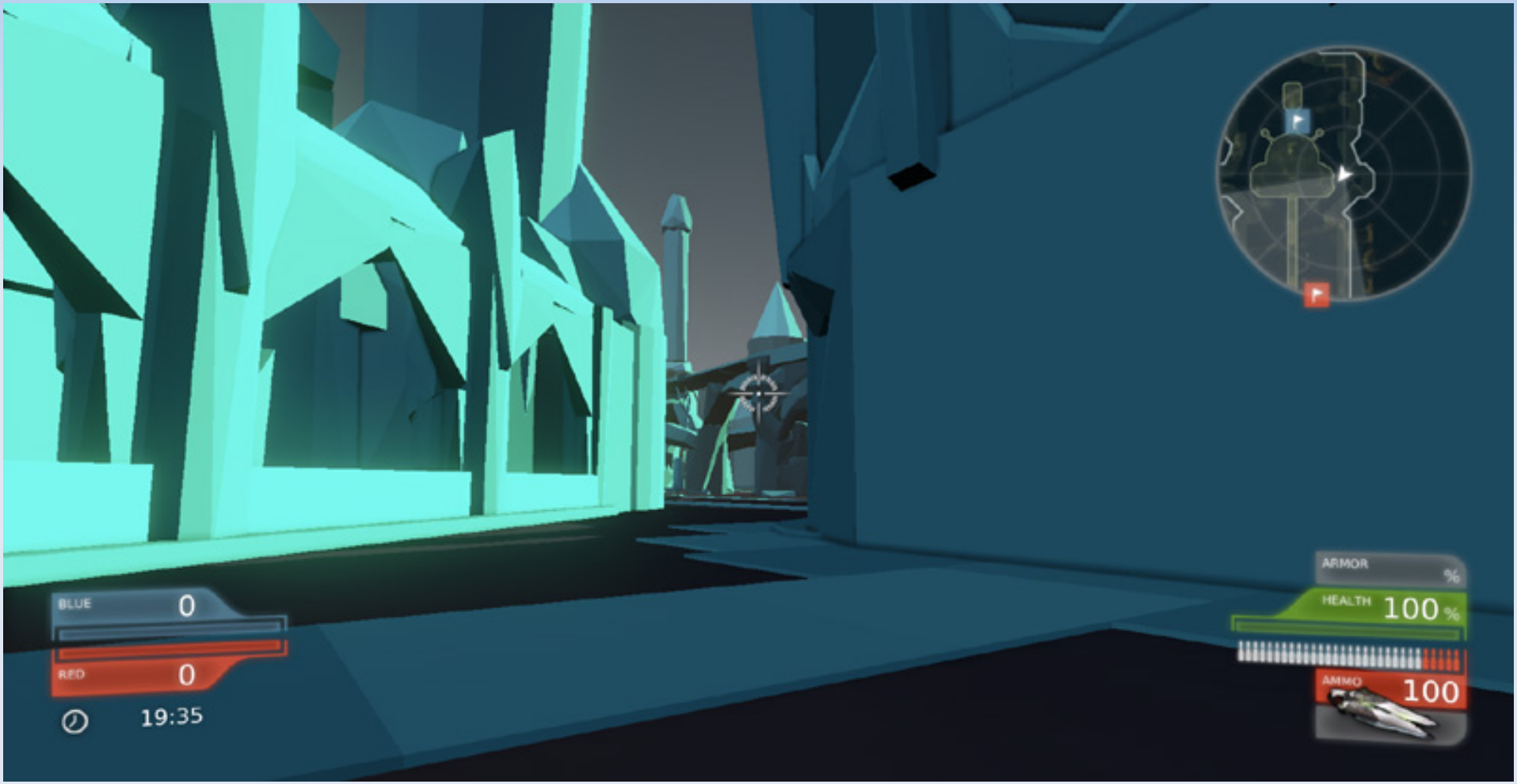
- All objects can collide with all other objects, collision detection is O(n^2)
- Avoid doing complicated physics collisions, use simplified collision geometry.

- Note: Does not care about motion. Timesteps are kept small enough for it to work anyway

Physics – Collision Detection



Physics – Collision Detection



Physics – Collision Detection

- Common basic shapes are:

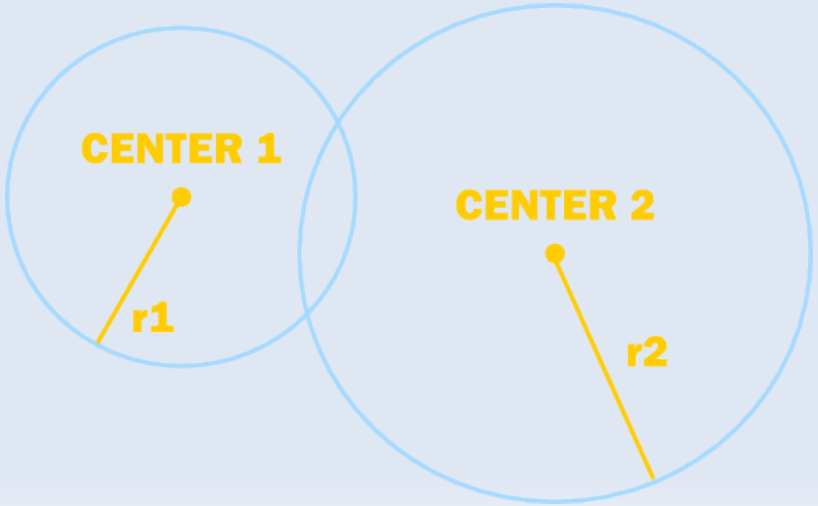
2D	**3D**
Circles	Spheres
AABB	Boxes
Convex Polygon	Convex Polyhedron

- Advanced shapes
  - Pixel perfect (2D)
  - Heightmaps (2D / 3D)

Circle / Circle

- Simplest and quickest

- Two circles intersect iff their centers are closer than the sum of their radii.
- SQ(c1[0]-c2[0]) + SQ(c1[1]-c2[1]) < SQ(r1+r2)



AABB / AABB

- Two convex objects do not overlap if there exists a line onto which the two objects' projections do not overlap.
- If two convex polygons do not intersect, then one of the two polygons' sides is the separating axis.

- Together, these imply we can search through all potential separating axes and see if we find a solution!

AABB / AABB

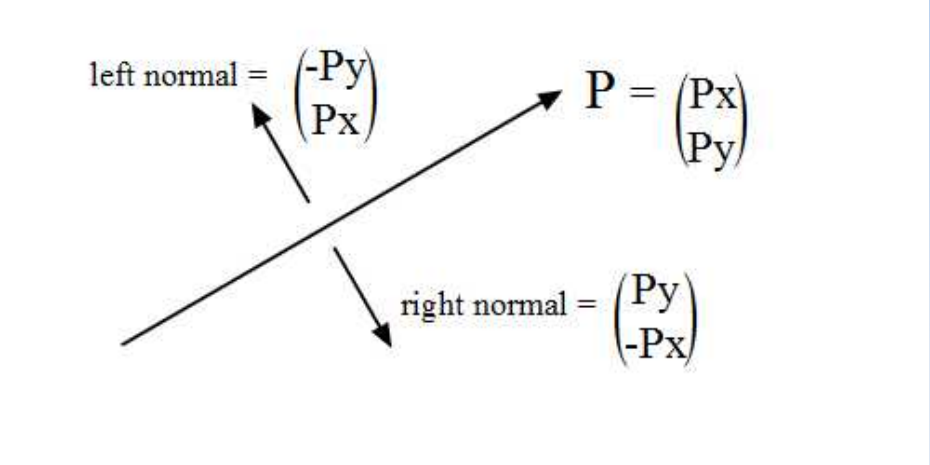
- Separating axis must be X-axis or Y-axis
- Check in order:
  - Is box1 left of box2?
  - Is box1 right of box2?
  - Is box1 above box2?
  - Is box1 below box2?

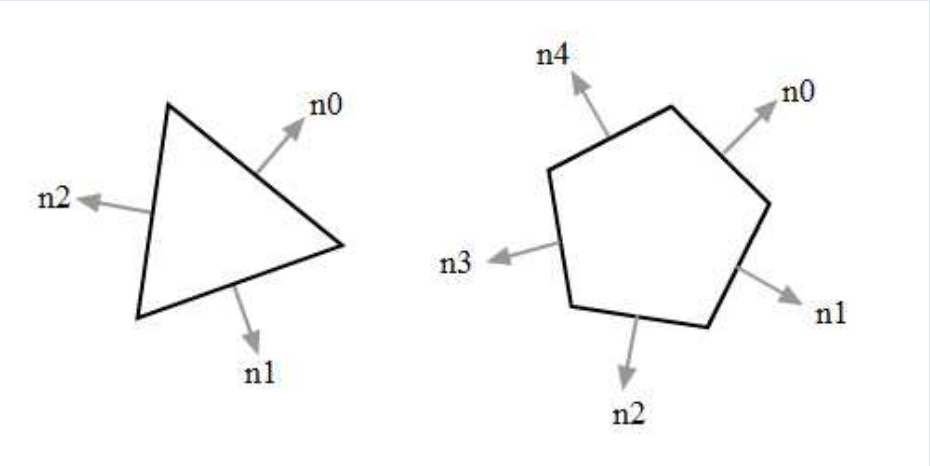
- Same check that we did for the camera!

Polygon / Polygon

Polygon / Polygon

- Create vectors for each side, see if all points of one polygon are on the “outside” half of the world.
  - These are called normal vectors





Polygon / Polygon

- If the last slide didn't convince you, polygon / polygon collision is SLOW and COMPLICATED
- Avoid it where possible!
- It is easier and faster to do pixel-perfect collision detection than polygon collision detection.

- BONUS:
  - This also solves AABB / Polygon, Point / Polygon, and Circle / Polygon